Introduction to Human Language Technologies

9. Syntactic parsing: parsers
Outline

1. Syntactic parsers
   - Background
   - Chart-based methods

2. CKY algorithm
Outline

1. Syntactic parsers
   - Background
   - Chart-based methods

2. CKY algorithm
Factors in parsing

Parsing performance depends on many aspects:

- Grammar expressivity (combination of symbols)
- Coverage (words)
- Parsing strategy (bottom-up, top-down)
- Rule application order (largest rule, most likely rule)
- Ambiguity management (keep all, select one - probabilities, semantics, pragmatics)
- ...
The problem of repeating derivations

- Top-down and bottom-up strategies both lead to repeated derivations when using backtracking.

Ex: "a flight from Indianapolis to Houston [on TWA...]

NG → NN
NG → NG PP

```
NP
  / \   
DT   NG
  /   / 
 a  NG  PP
     / 
    NN from Indianapolis
     |
  flight
```
The problem of repeating derivations

- Top-down and bottom-up strategies both lead to repeated derivations when using backtracking.
  
  Ex: "a flight from Indianapolis to Houston"

  NG → NN
  NG → NG PP

  NP
  └── NG
     ├── DT
     │   └── a
     └── NG
           ├── NG
           │   └── flight
           └── PP
               └── to Houston

           └── PP
               └── from Indianapolis
Outline

1. Syntactic parsers
   - Background
   - Chart-based methods

2. CKY algorithm
Properties

- They avoid re-doing derivations using dynamic programming.
- They represent derivations as a directed graph named chart.
- They use a dynamic programming table to build the chart.
Chart

- Nodes: positions between words of the input sentence
- Edges: dotted rules subsuming a sequence of words of the input sentence

Dotted rules represent rules states:
- Passive rules: $A \rightarrow B_1 \ldots B_k \bullet$
- Active rules: $A \rightarrow B_1 \ldots B_i \bullet B_{i+1} \ldots B_k$

Ex:

```
NP \rightarrow \bullet det \ n
```

```
NP \rightarrow det \ n \bullet
```

```
NP \rightarrow det \n \bullet
```

```
NP \rightarrow vi \bullet
```

1 \ the \ 2 \ cat \ 3 \ eats \ 4 \ fish \ 5
Chart as a dynamic programming table

<table>
<thead>
<tr>
<th></th>
<th>NP → _ det n</th>
<th>NP → det n _</th>
<th>NP → det n</th>
<th>NP → vi _</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>_ det n</td>
<td>det n _</td>
<td>det n</td>
<td>vi _</td>
</tr>
<tr>
<td>2</td>
<td>the</td>
<td>cat</td>
<td>eats</td>
<td>fish</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chart-based methods

Syntactic parsers

CKY algorithm
Popular chart-based algorithms

- **CKY algorithm** (Younger, 1967)
  - introduced dynamic programming
  - limited to CFGs in Chomsky Normal Form
  - passive bottom-up chart parser (only passive rules)
  - straightforward probabilistic version

- **Earley algorithm** (Earley, 1970)
  - any CFG
  - active top-down parser (active/passive rules)
  - non-straightforward probabilistic version

- **Generalized chart parsing** (Kay, 1980)
Outline

1. Syntactic parsers
   - Background
   - Chart-based methods

2. CKY algorithm
A CFG $G = (N, \Sigma, R, S)$ expressed in CNF is as follows:

- $N$ is a set of non-terminal symbols
- $\Sigma$ is a set of terminal symbols
- $R$ is a set of rules which take one of two forms:
  - $X \rightarrow Y_1 Y_2$ for $X, Y_1, Y_2 \in N$
  - $X \rightarrow \alpha$ for $X \in N$ and $\alpha \in \Sigma$
- $S \in N$ is a start symbol

Any CFG can be converted into CNF
CNF conversion

1. Convert Hybrid rules: replace terminals with new non-terminals
   
   **Ex:** \( INF_{-}VP \rightarrow to \ VP \Rightarrow \)
   
   \( INF_{-}VP \rightarrow TO \ VP \)
   
   \( TO \rightarrow to \)

2. Convert non-binary rules:
   
   **Ex:** \( S \rightarrow VP \ NP \ PP \Rightarrow \)
   
   \( S \rightarrow VP \ X \)
   
   \( X \rightarrow NP \ PP \)

3. Convert unit productions: \( A \rightarrow^{*} B \) and \( B \rightarrow \alpha \Rightarrow A \rightarrow \alpha \)
   
   **Ex:** \( NP \rightarrow N \) and \( N \rightarrow dog \Rightarrow NP \rightarrow dog \)
Exercise

Covert the following CFG to CNF

1. \( S \rightarrow NP \ VP \)
2. \( NP \rightarrow det \ n \)
3. \( NP \rightarrow n \)
4. \( VP \rightarrow vt \ NP \ PP \)
5. \( VP \rightarrow vi \)
6. \( PP \rightarrow with \ NP \)
7. \( det \rightarrow the | a \)
8. \( n \rightarrow cat | fish | knife \)
9. \( vt \rightarrow eats \)
10. \( vi \rightarrow eats \)
CKY Algorithm

Chart content:

- Maximum probability of a subtree with root $X$ spanning words $i \ldots j$:
  \[ \pi(i, j, X) \]

- Backpath to recover which rules produced the maximum probability tree:
  \[ \psi(i, j, X) \]

The goal is to compute:

- \( \max_{t \in \mathcal{T}(s)} p(t) = \pi(1, n, S) \)
- \( \psi(1, n, S) \)
- It is possible to use it without probabilities to get all parse trees (with higher complexity)
CKY Algorithm

Base case: Tree leaves
- $\forall i = 1 \ldots n, \forall X \rightarrow w_i \in R, \pi(i, i, X) = q(X \rightarrow w_i)$

Recursive case: Non-terminal nodes
- $\forall i = 1 \ldots n, \forall j = (i + 1) \ldots n, \forall X \in N$
  $$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, k:i \leq k < j} q(X \rightarrow YZ) \times \pi(i, k, Y) \times \pi(k + 1, j, Z)$$

  $$\psi(i, j, X) = \arg \max_{X \rightarrow YZ \in R, k:i \leq k < j} q(X \rightarrow YZ) \times \pi(i, k, Y) \times \pi(k + 1, j, Z)$$

Output:
- Return $\pi(1, n, S)$ and recover backpath through $\psi(1, n, S)$
CKY Algorithm

Suppose $s = w_1w_2w_3w_4$ and $G = \langle N, \Sigma, S, R, q \rangle$ a PCFG

$R = \{ X_k \rightarrow Y_s Z_t \} \cup \{ X_k \rightarrow \alpha \}$
CKY Algorithm

Suppose $s = w_1 w_2 w_3 w_4$ and $G = < N, \Sigma, S, R, q >$ a PCFG

$R = \{ X_k \rightarrow Y_s Z_t \} \cup \{ X_k \rightarrow \alpha \}$

**Base case:**
$\forall_k \text{ shot } \{ X_k \rightarrow w_i \}$ to compute $\pi(i, i, X_k)$
CKY Algorithm

Suppose $s = w_1w_2w_3w_4$ and $G = \langle N, \Sigma, S, R, q \rangle$ a PCFG

$R = \{ X_k \Rightarrow Y_s Z_t \} \cup \{ X_k \Rightarrow \alpha \}$

**Recursive case:**

$\forall k$ shot $\{ X_k \Rightarrow Y_s Z_t \}$ to get $\varphi(i, j, X_k)$ and compute $\pi(i, j, X_k)$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>${ X_k \Rightarrow w_1 }_k$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>${ X_k \Rightarrow w_2 }_k$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>${ X_k \Rightarrow w_3 }_k$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>${ X_k \Rightarrow w_4 }_k$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Suppose \( s = w_1 w_2 w_3 w_4 \) and \( G = \langle N, \Sigma, S, R, q \rangle \) a PCFG

\[
R = \{ X_k \rightarrow Y_s Z_t \} \cup \{ X_k \rightarrow \alpha \}
\]

**Recursive case:**

\( \forall k \) shot \( \{ X_k \rightarrow Y_s Z_t \} \) to get \( \varphi(i, j, X_k) \) and compute \( \pi(i, j, X_k) \)
CKY Algorithm

Suppose \( s = w_1 w_2 w_3 w_4 \) and \( G = \langle N, \Sigma, S, R, q \rangle \) a PCFG

\[ R = \{ X_k \rightarrow Y_s Z_t \} \cup \{ X_k \rightarrow \alpha \} \]

**Recursive case:**

\( \forall_k \) shot \( \{ X_k \rightarrow Y_s Z_t \} \) to get \( \varphi(i, j, X_k) \) and compute \( \pi(i, j, X_k) \)

---

**Example for (1,3):**

\( \{ X_k \rightarrow w_1 \}_k \)  \( \{ X_k \rightarrow w_2 \}_k \)  \( \{ X_k \rightarrow w_3 \}_k \)  \( \{ X_k \rightarrow w_4 \}_k \)
CKY Algorithm

Suppose $s = w_1 w_2 w_3 w_4$ and $G = \langle N, \Sigma, S, R, q \rangle$ a PCFG

$R = \{ X_k \rightarrow Y_s Z_t \} \cup \{ X_k \rightarrow \alpha \}$

Recursive case:

$\forall k \text{ shot } \{ X_k \rightarrow Y_s Z_t \}$ to get $\varphi(i,j,X_k)$ and compute $\pi(i,j,X_k)$

Example for $(1,3)$

\[
\begin{array}{c|c|c|c|c}
  & 1 & 2 & 3 & 4 \\
\hline
11 & \{ X_k \rightarrow w_1 \}_k & 12 & 13 & \{ X_k \rightarrow Y_s,11 Z_{t,23} \}_k \\
\hline
22 & \{ X_k \rightarrow w_2 \}_k & 23 & 24 & \\
\hline
33 & \{ X_k \rightarrow w_3 \}_k & & & \\
\hline
44 & \{ X_k \rightarrow w_4 \}_k & & & \\
\end{array}
\]
CKY Algorithm

Suppose $s = w_1 w_2 w_3 w_4$ and $G = \langle N, \Sigma, S, R, q \rangle$ a PCFG

$R = \{X_k \rightarrow Y_s Z_t\} \cup \{X_k \rightarrow \alpha\}$

**Recursive case:**

$\forall k \text{ shot } \{X_k \rightarrow Y_s Z_t\}$ to get $\varphi(i, j, X_k)$ and compute $\pi(i, j, X_k)$

Example for $(1,3)$

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>${X_k \rightarrow w_1}_k$</td>
<td>${X_k \rightarrow w_2}_k$</td>
<td>${X_k \rightarrow w_3}_k$</td>
<td>${X_k \rightarrow w_4}_k$</td>
</tr>
<tr>
<td>11</td>
<td>22</td>
<td>33</td>
<td>44</td>
</tr>
<tr>
<td>13</td>
<td>23</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>${X_k \rightarrow Y_{s,12} Z_{t,33}}_k$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CKY Algorithm

**Input:** a sentence $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.

**Initialization:**
For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

**Algorithm:**
- For $l = 1 \ldots (n - 1)$
  - For $i = 1 \ldots (n - l)$
    * Set $j = i + l$
    * For all $X \in N$, calculate
      $$\pi(i, j, X) = \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

and

$$bp(i, j, X) = \arg \max_{X \rightarrow YZ \in R, \ s \in \{i \ldots (j-1)\}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s + 1, j, Z))$$

**Output:** Return $\pi(1, n, S) = \max_{t \in T(s)} p(t)$, and backpointers $bp$ which allow recovery of $\arg \max_{t \in T(s)} p(t)$. 

Exercise

Compute the best parse tree and its probability for the following input sentence using the PCFG:

“the woman saw the man with the telescope”

S → NP VP 0.5    Vi → sleeps 1.0
S → NP Vi 0.5    Vt → saw 1.0
NP → DT NN 0.4    NN → man 0.7
NP → NP PP 0.6    NN → woman 0.2
PP → IN NP 1.0    NN → telescope 0.1
VP → Vt NP 0.4    DT → the 1.0
VP → VP PP 0.1    IN → with 0.5
VP → Vi PP 0.5    IN → in 0.5